

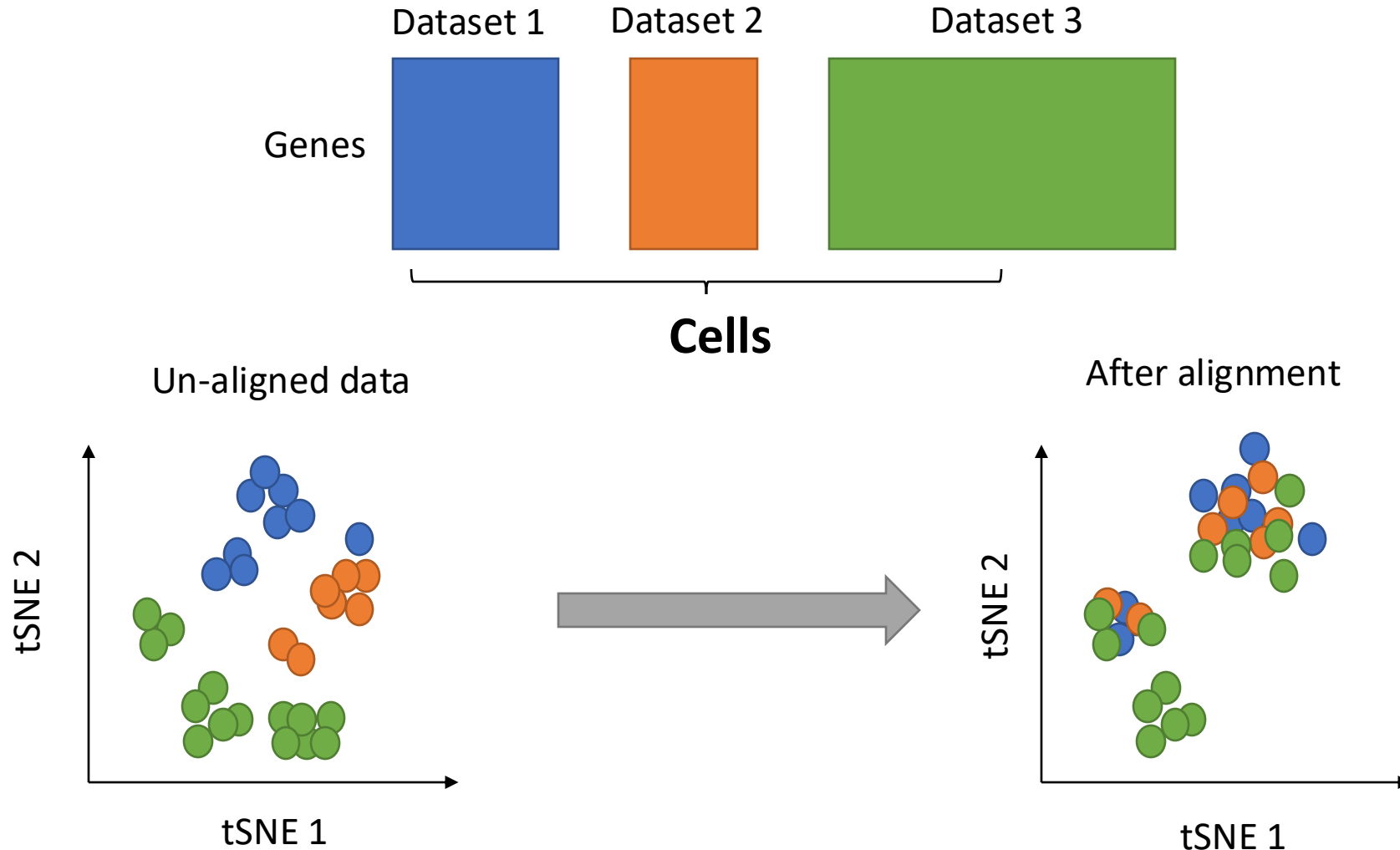
Lecture 9

Data integration and batch correction

Outline

- scRNA-seq data integration and batch correction
 - Three types of integration for single-cell multi-omics data
 - Factor model-based methods
 - Linear models
 - Variational autoencoders
 - Cell-similarity based methods
 - Comparison between different methods

What is data integration/alignment?

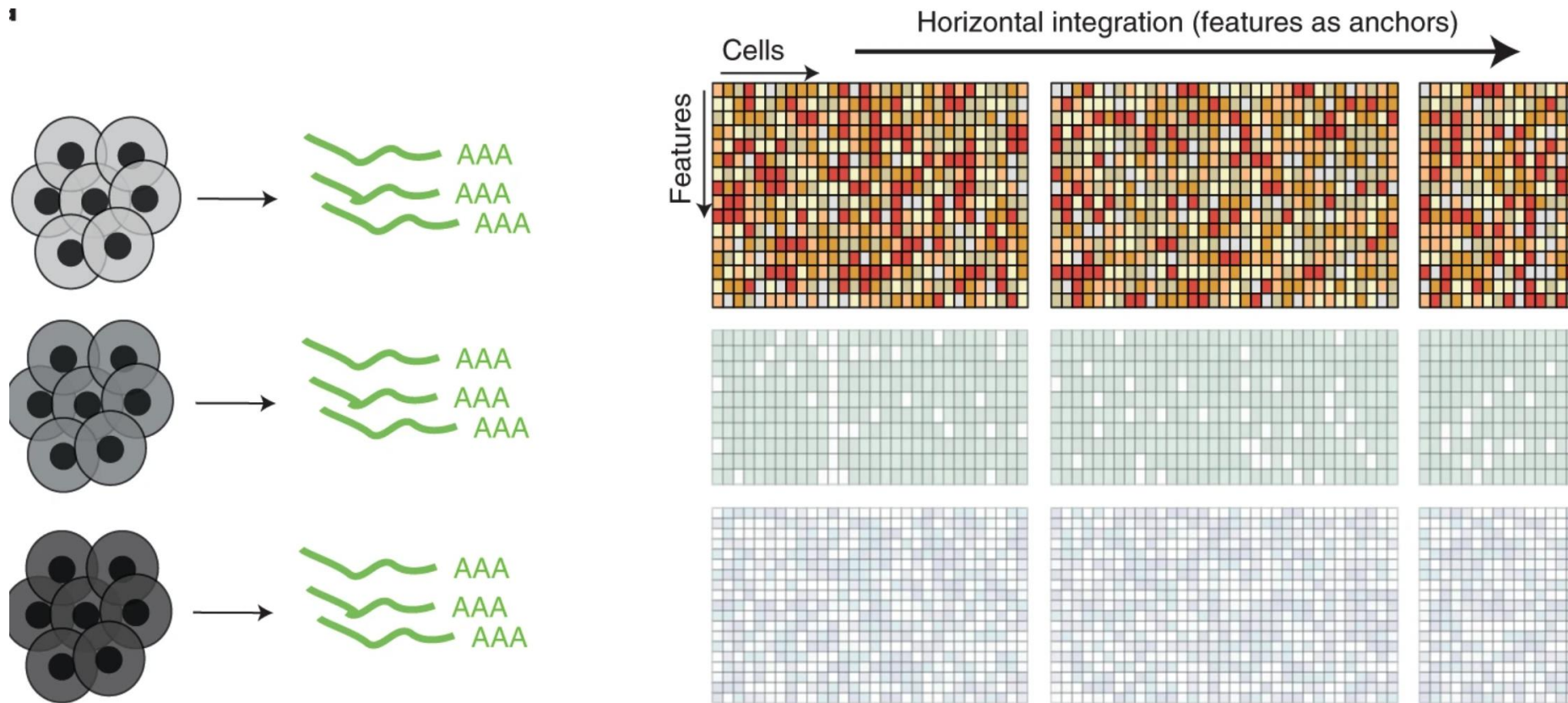


- Data integration may serve as the first step before any down-stream analyses
 - Double dipping: cells are not longer independent anymore after integration
- Observations are no longer counts, or only obtain low-dimensional features

Three types of data integration for single-cell multiomics data

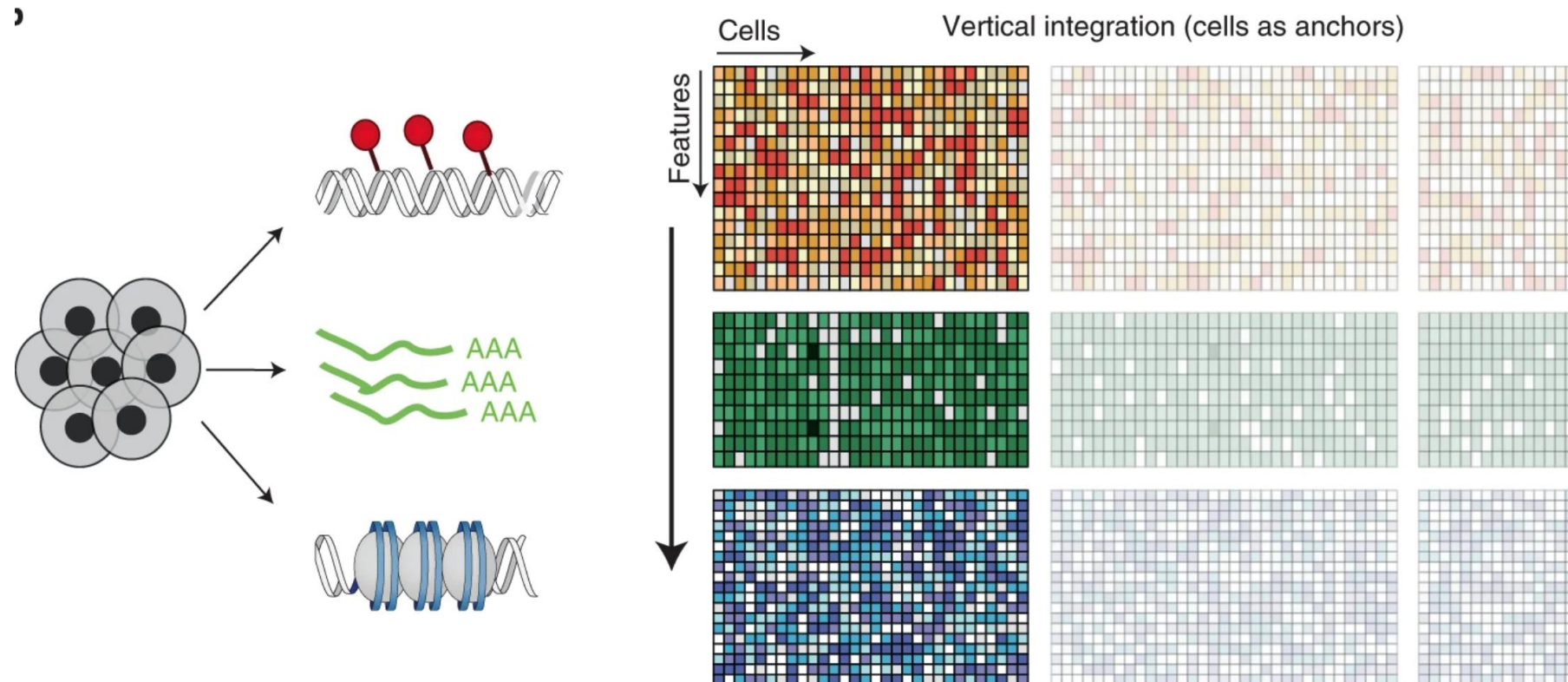
[Argelaguet et. al., Nature Biotech 2021]

- Same sets of features, different datasets
 - Main challenge: batch effects



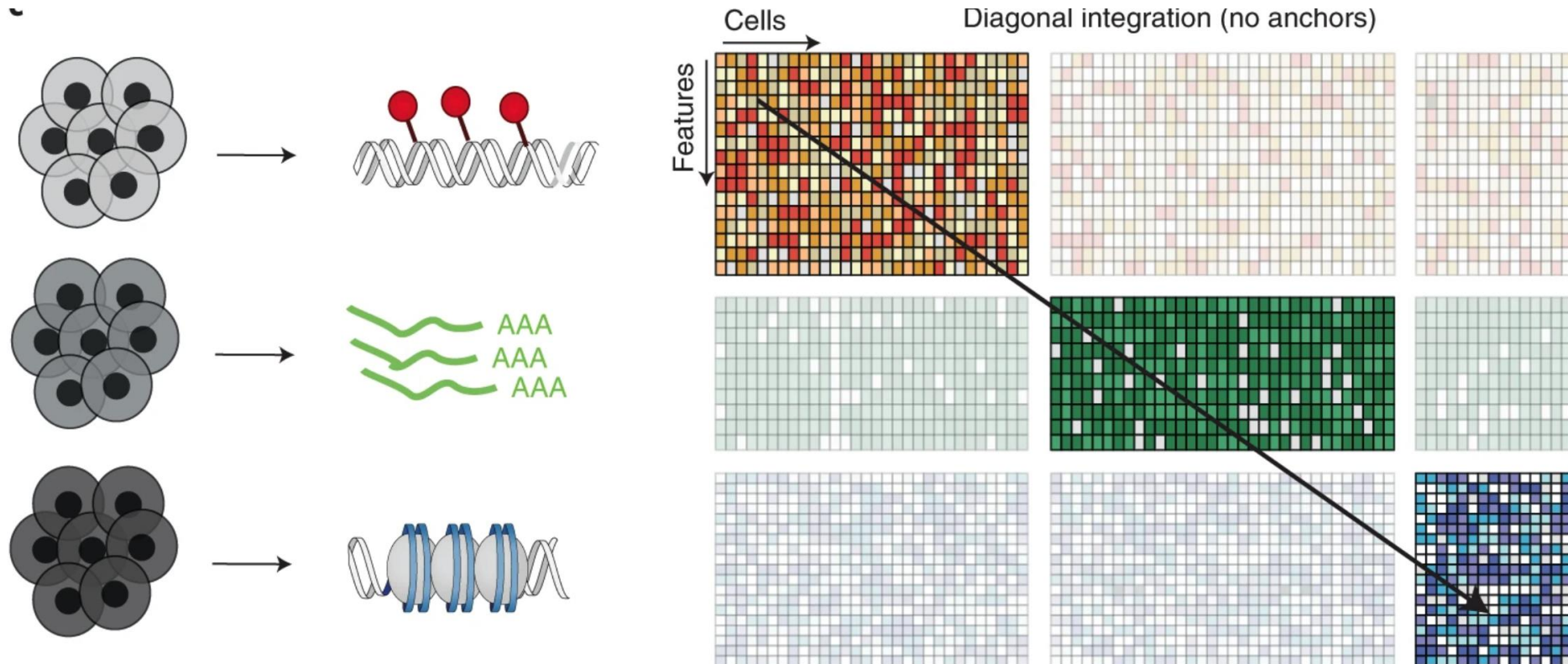
Three types of data integration for single-cell multiomics data

- Same cell, different types of features (multimodal data)
 - Combine different types of features to understand cell-cell similarity
 - Missing modality in some datasets



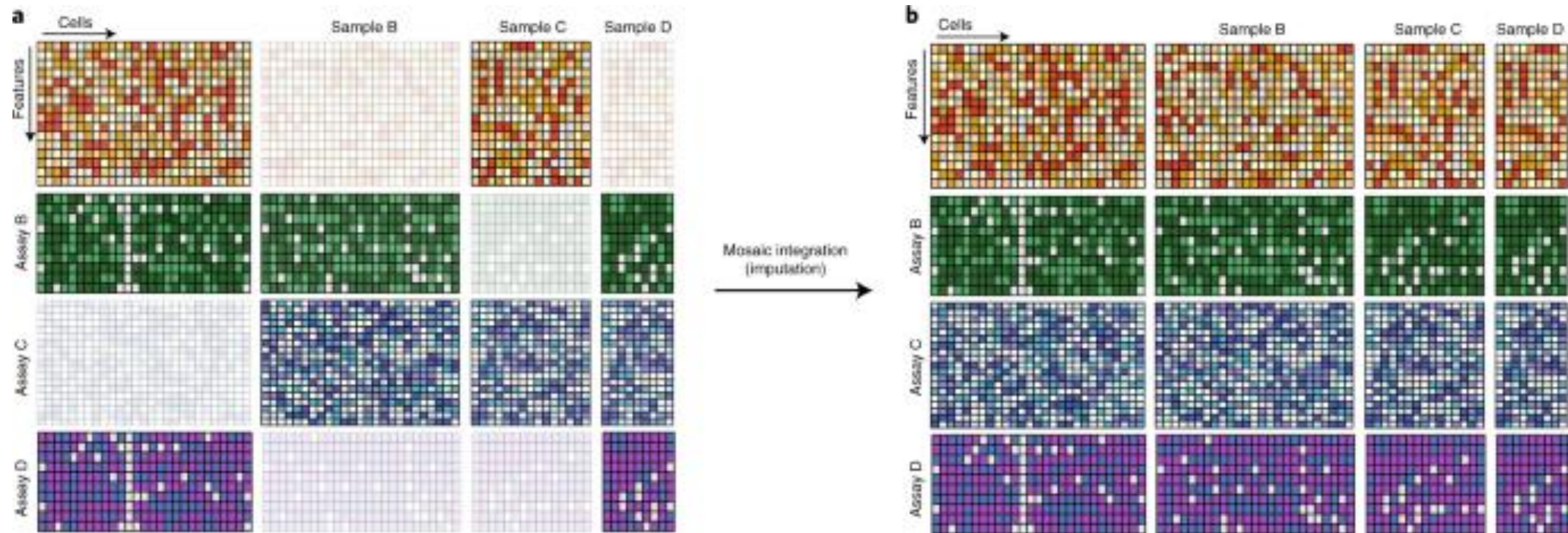
Three types of data integration for single-cell multiomics data

- Different cells, different types of features
 - What is the basis for integration?
 - Extra information about feature connections
 - Use subset of cells with overlapping features as “bridges”



Three types of data integration for single-cell multiomics data

- Mosaic integration between the second and third types



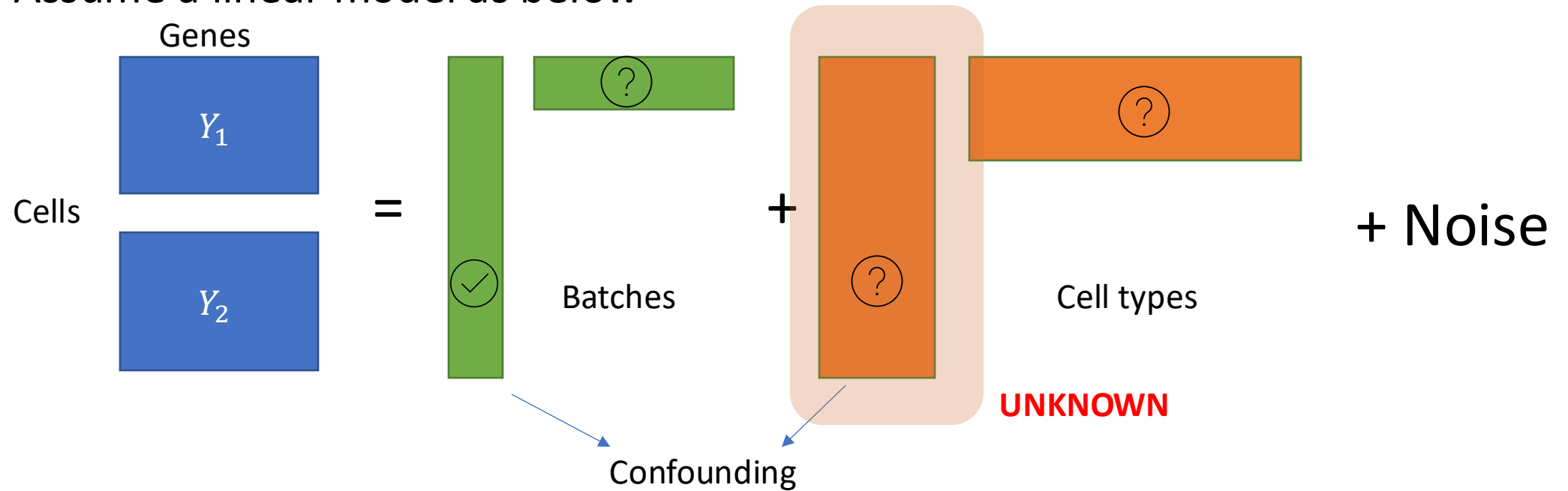
Integration for scRNA-seq data = batch correction?

Un-alignment between datasets

- Biological differences:
 - Different cell population (tissue, individual, species)
 - Different cell types
- Technical differences:
 - batch effects
 - different sequencing depth
- Jointly analyze of multiple datasets
 - Remove batch effects
 - Remove unwanted/not interesting biological differences
 - ‘uninteresting’ differences between individuals, species
 - Keep meaningful biological difference between datasets (such as new cell type or true differential expression of cell type marker genes between conditions)
- Challenge: “unknown” Confounding between batches and cell types

Unsupervised Batch Effect Removal

- Confounding between batches and unknown cell types
 - Assume a linear model as below



- $X\beta^T + UV^T = X(\beta + \gamma)^T + (UV^T - X\gamma^T)$
- Batches can be confounded with other important biological signals
 - Batch effects may not be identifiable without additional assumptions (for instance: $X \perp U$)

Batch correction with linear model: Limma (Ritchie et. al. , NAR 2015)

- The overall gene expression matrix (mean matrix): $\mu_g \times 1_n^T$
- Batch corrected data: $X - R_s D_s$
- Developed for bulk RNA-seq data where differential testing across conditions is the primary goal
- (D_c^T, D_s^T) needs to be full rank. For scRNA-seq, conditions and batches can be perfectly confounded
- Batches can also be confounded with cell types, trajectories in scRNA-seq

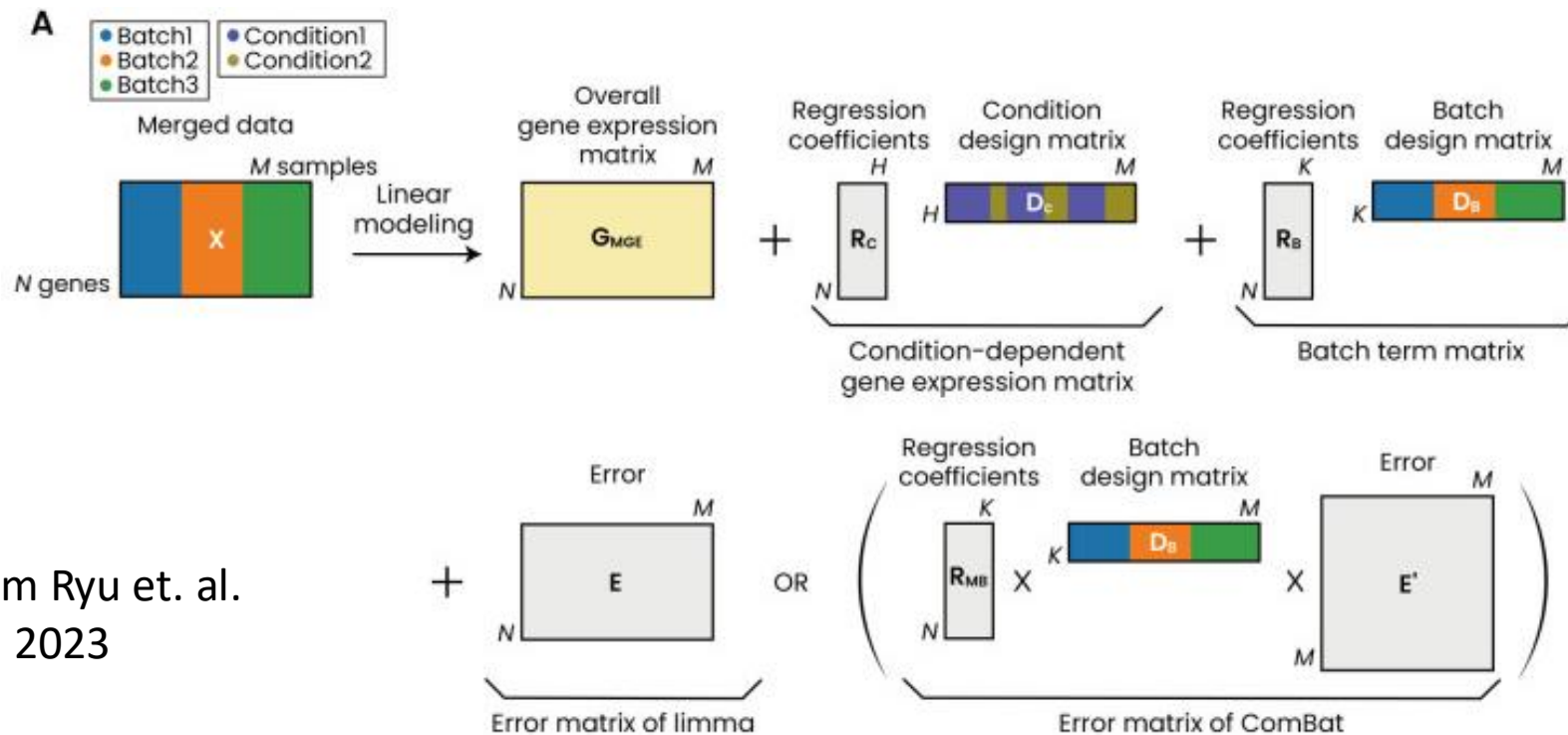
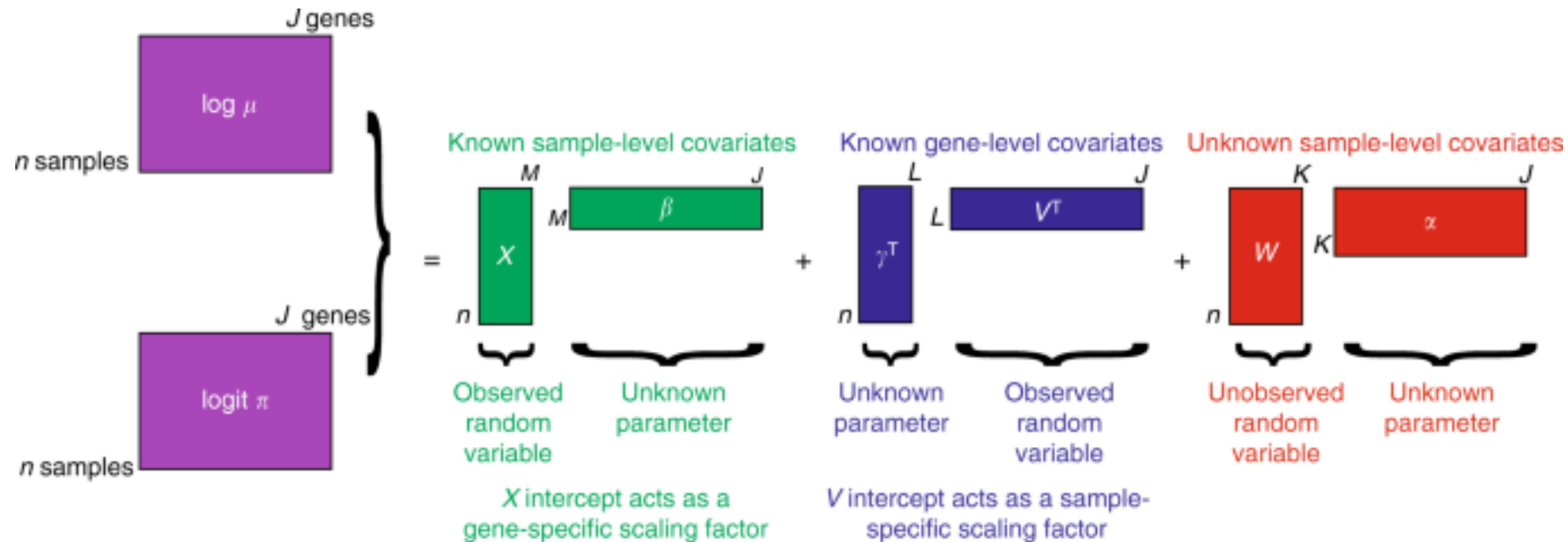


Figure from Ryu et. al.
Mol Cells, 2023

Challenges for batch correction in scRNA-seq

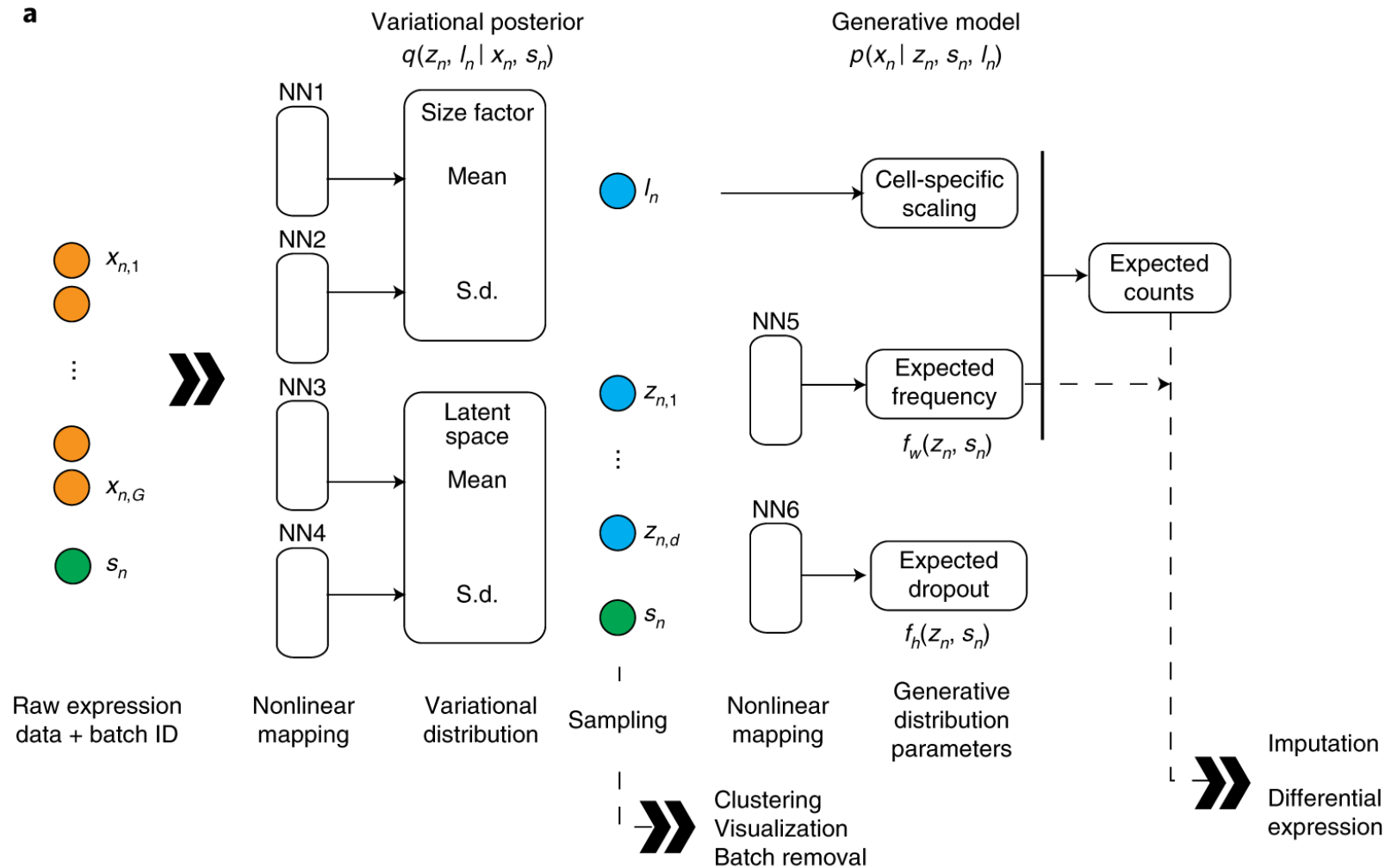
- Batch effects may not be linear
- If Batches are confounded with hidden factors of the data (like clustering structure), then batch effects are not identifiable
 - $Y_{\text{cells*genes}} = ZV^T + X_{\text{batch}}\beta + \text{error}$, β is not identifiable if the latent factors U and X_{batch} can be arbitrarily correlated
 - One possible identifiability condition: within the same cell type, cells are biologically homogenous across batches
 - If cell types are already known, what is the purpose of integration?
 - Another possible identifiability condition (implicitly assumed in many similarity based methods): $X_{\text{batch}}\beta$ is small compared to ZV^T , similar cells in batch 2 to a cell in batch 1 keep the same with/without batches
- Current batch correction methods tend to overcorrect batches effects (Argelaguet et. al., Nature Biotech 2021). Differential testing between conditions may tend to be conservative after correction

ZINB-WaVE (Risso et. al., Nature Comm 2018)



- The batches are sample-level covariates
- Gene-level covariates can include gene features such as gene length and GC content
- W is the batch adjusted latent representation of cells
- Implicitly assume that the latent factors and batches are uncorrelated
 - They used an L2 penalization on W in the loss function (equivalent to independent Gaussian prior on W)
 - The algorithm does not force latent factors and batches to be uncorrelated
- Assume that the observed counts follow ZINB model

scVI (Lopez et. al. Nature Methods 2018)



- Use variational autoencoder (next page)
- Main feature: add batch information as extra nodes in both the input and bottleneck layer
- Implicitly assumes that latent factors Z and X_{batch} are uncorrelated as X_{batch} is fixed and Z has prior $Z \sim N(0, I)$
- Under linear model

$$Y_{cells*genes} = UV^T + X_{batch}\beta + E$$
 estimated Z and X_{batch} are uncorrelated
- Estimated Z can be correlated with X_{batch} in scVI because of using VAE

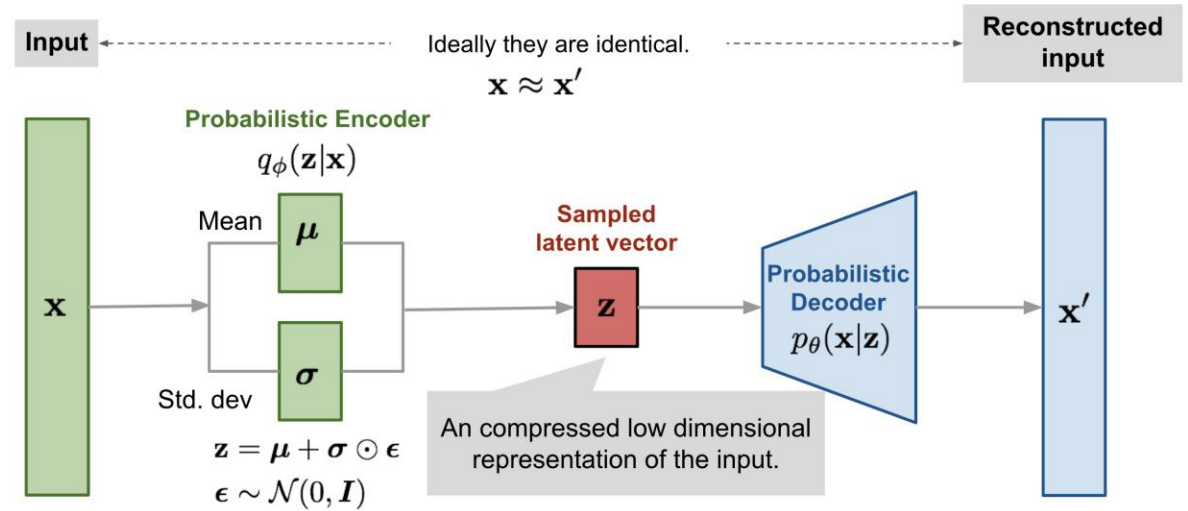
Details of scVI model

Variational autoencoder

- Assume that the latent variables $Z \sim N(0, I)$
- Approximate the posterior of Z given input data by Gaussian distribution
- Encoder: posterior mean and variance of Z as non-linear functions of input data
- Decoder: non-linear mapping from Z to the observed data
- Generalization of linear probabilistic factor model to nonlinear probabilistic factor models
- scVI assumes a ZINB model on the observed data
 - Both posterior distributions of Z and mapping from Z to the observed data depend on the batches

Final output of scVI

- Use latent factors for visualization and clustering
- Use output layer for denoising (imputation)



$$\begin{aligned}
 z_n &\sim \text{Normal}(0, \mathbf{I}) \\
 \ell_n &\sim \text{log normal}(\ell_\mu, \ell_\sigma^2) \\
 \rho_n &= f_w(z_n, s_n) \\
 w_{ng} &\sim \text{Gamma}(\rho_n^g, \theta) \\
 y_{ng} &\sim \text{Poisson}(\ell_n w_{ng}) \\
 h_{ng} &\sim \text{Bernoulli}(f_h^g(z_n, s_n)) \\
 x_{ng} &= \begin{cases} y_{ng} & \text{if } h_{ng} = 0 \\ 0 & \text{otherwise} \end{cases}
 \end{aligned}$$

scGen (Lotfollahi et. al. Nature Methods, 2019)

- Originally designed to perturbation prediction but can also be used for batch correction
- scGen also used VAE, not sure if batches are inputs in the VAE as in scVI
- Batch correction is done to each cell type separately
 - Requires cell type information as input data (may not be applicable in practice)
 - In the latent space, for each cell type, calculate
$$\delta = \text{avg}(z_{\text{condition}=1}) - \text{avg}(z_{\text{condition}=0})$$
 - Add δ to the corresponding latent vectors so cells within the same cell type are mixed
 - Get the corrected gene expression matrix

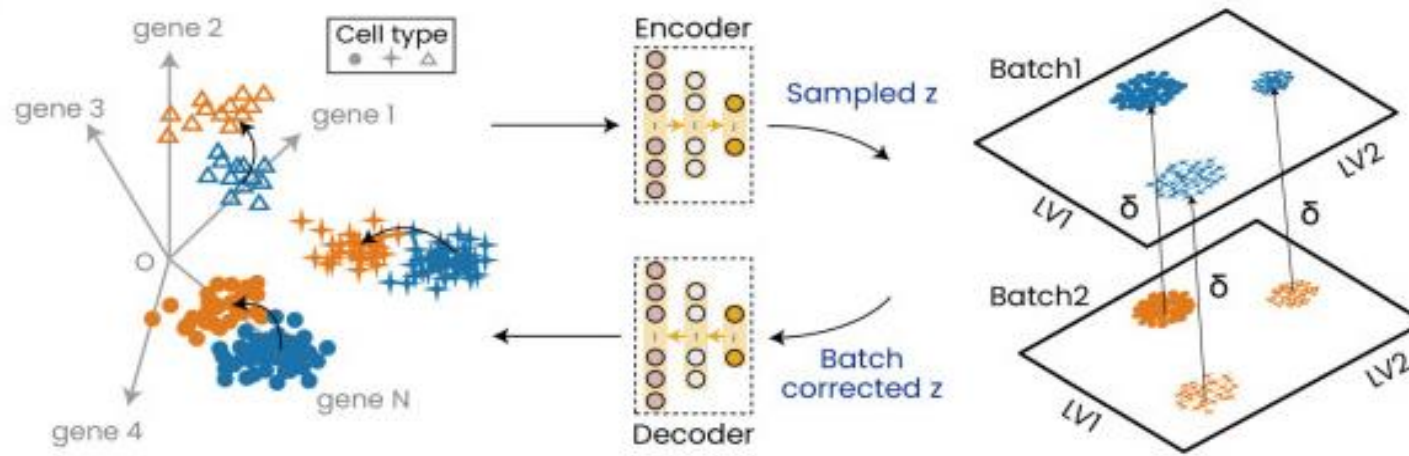
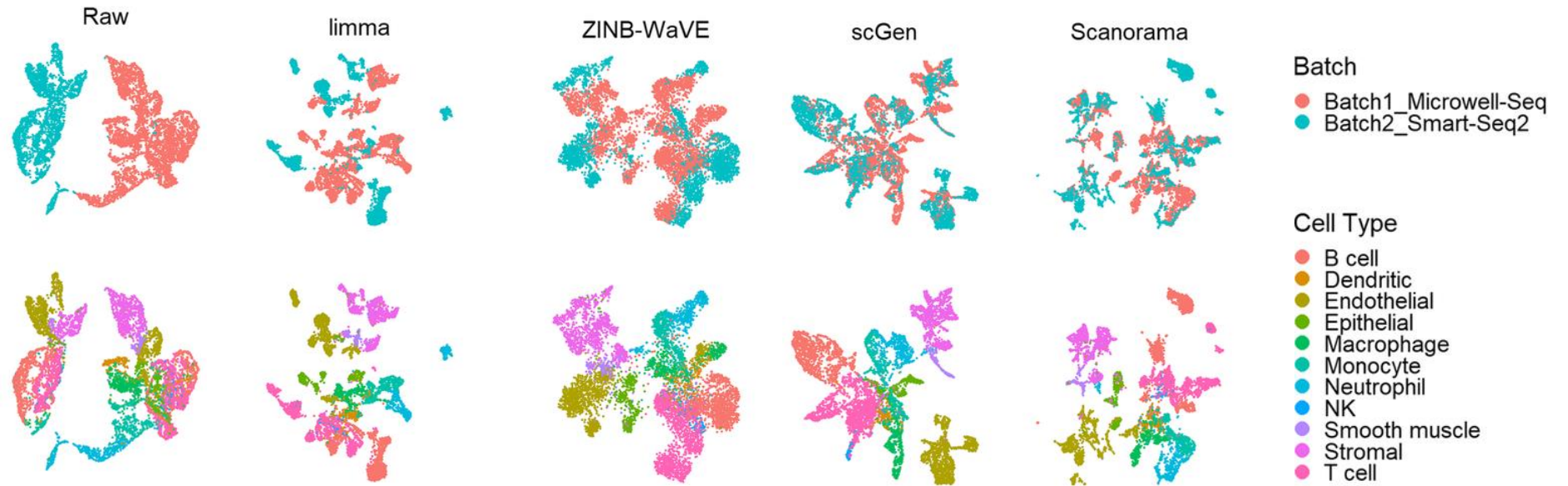


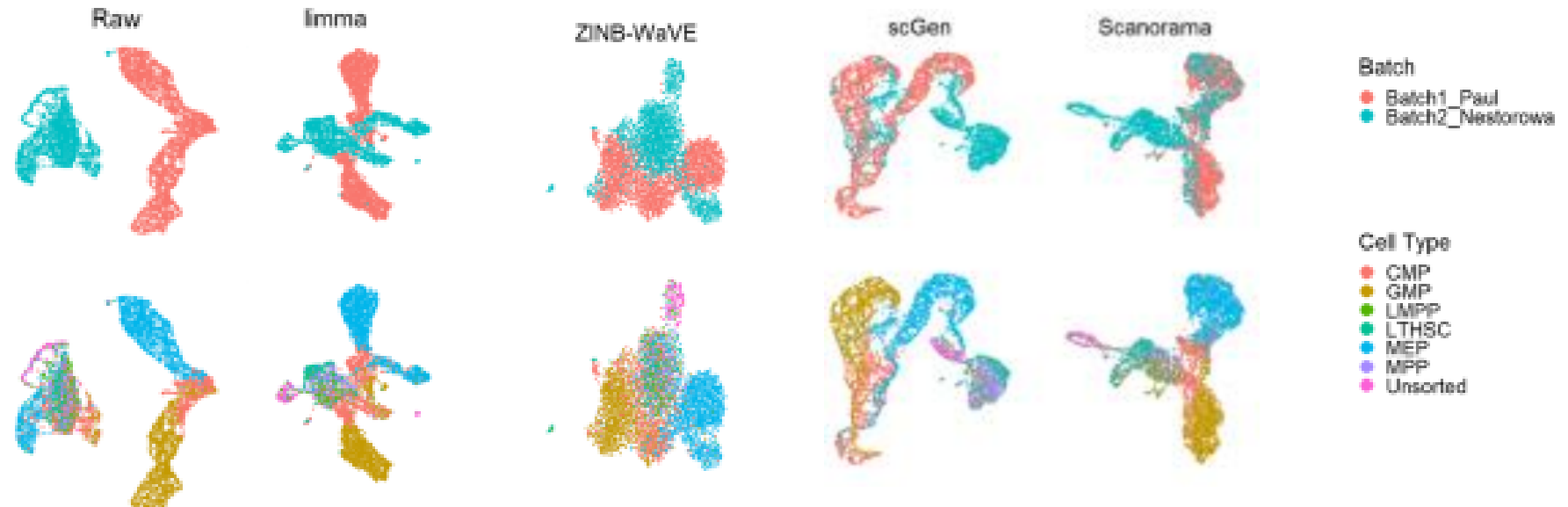
Figure from Ryu et. al. Mol Cells, 2023

Some benchmarking results (Tran et. al. Genome Biology 2018)

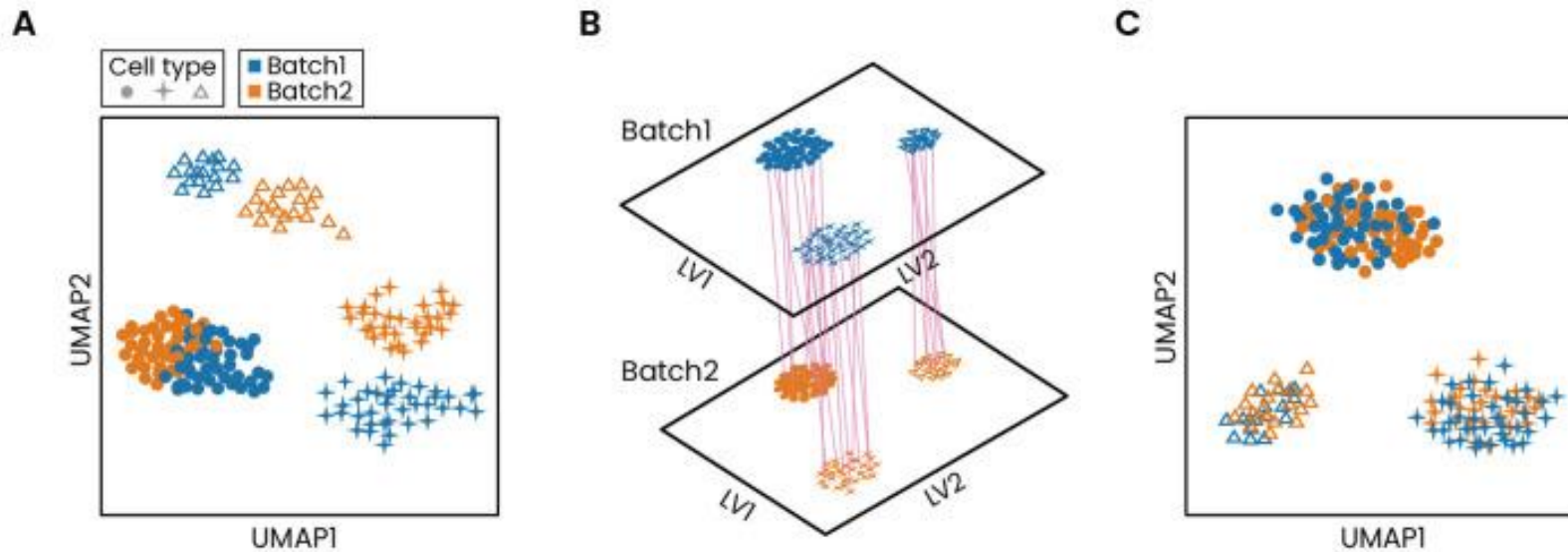
Identical cell
types, different
technologies



Non
identical cell
types



Similarity-based batch correction methods



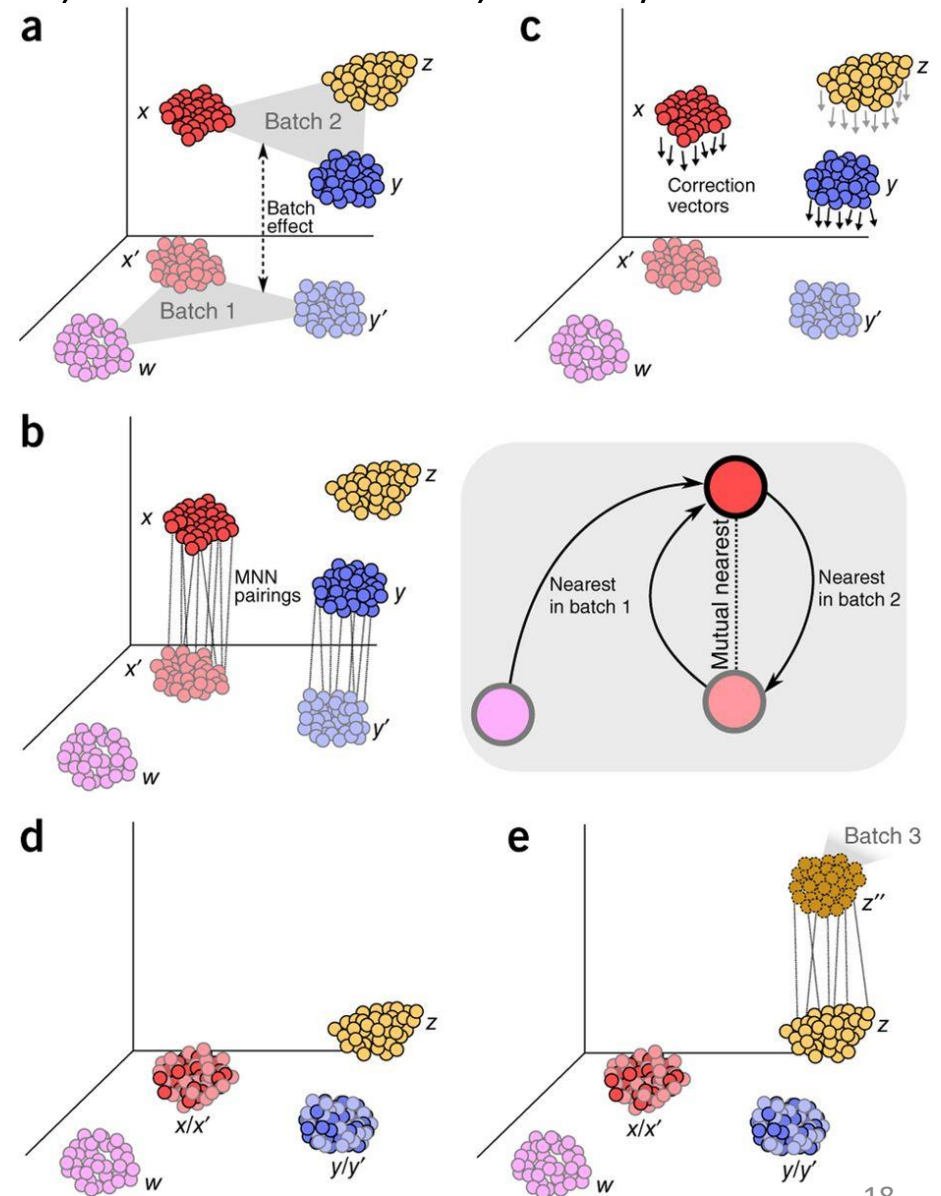
Common steps:

- Project the merged datasets onto a low-dimensional space
- Identify similar cells (pairs of cells) between batches
- Correction: correct batch effects so that cells pairs are together on the low-dimensional space
- Batch correction is only performed on low-dimensional space
 - Previous factor-model-based methods provide batch corrected gene expressions

MNNcorrect / fast MNN (Hadhverdi L. et. al., Nature Biotech, 2018)

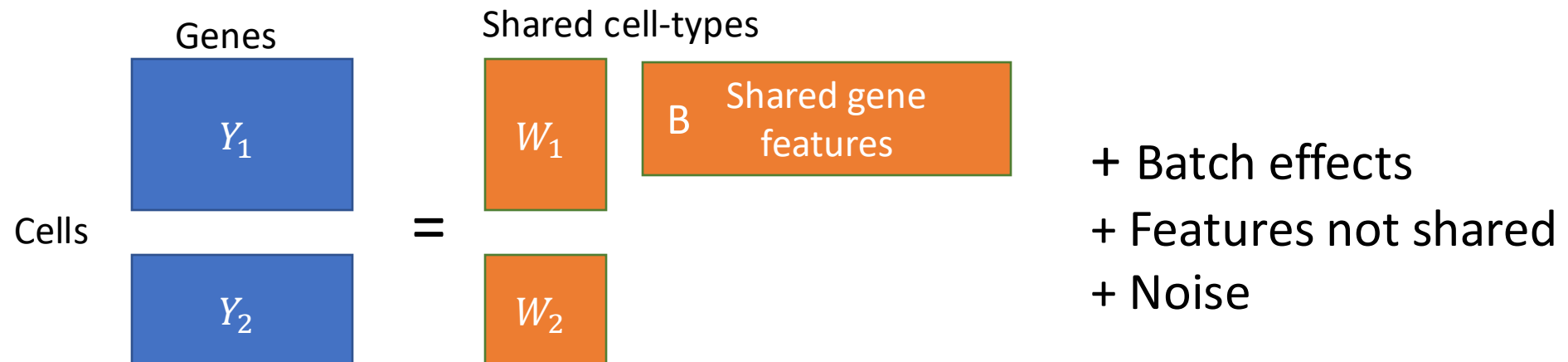
- Steps:

- Measure cell similarity (Euclidean distance after normalization)
 - Find paired cells from two batches
 - Identify KNN of each cell in batch 1 (2) in the other batch 2 (1)
 - Keep the pair of cells if they are both KNN of each other
 - Batch correction:
 - Compute pair-specific differences
 - Use Gaussian kernel smoothing (weights) to compute the correction vector of each cell
 - The cell-specific correction vector is a weighted average of the pair-specific correction vectors
- Critical assumption
 - Batch effects are relatively small



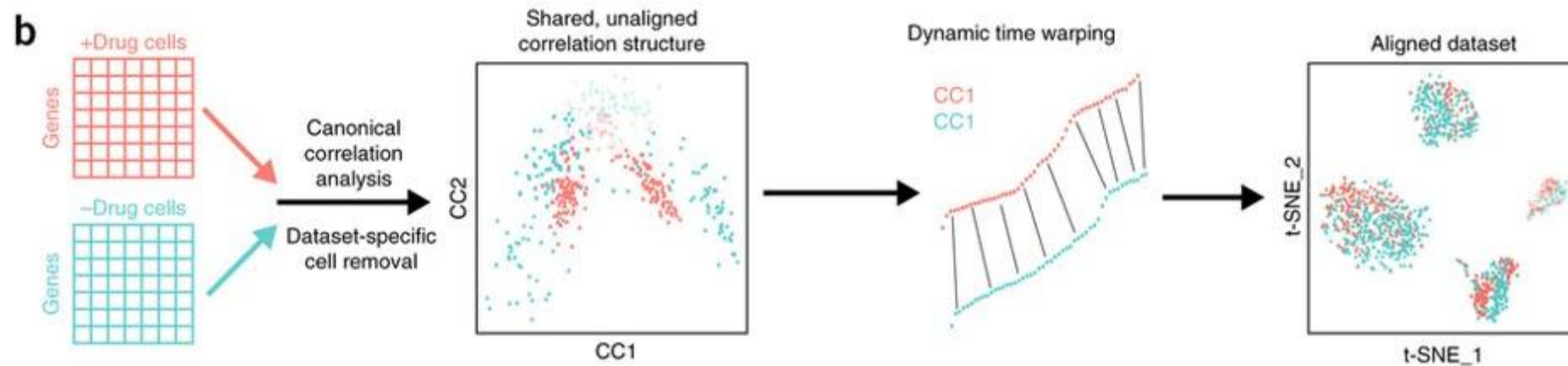
Use Canonical correlation analysis (CCA) for scRNA-seq alignment

- CCA: originally used to find best combination of two sets of variables that have largest correlation
 - For scRNA-seq, treat each cell as a “feature”, each gene as an “observation”
 - Compute weighted combination of cells within each batch so that the combined cells have best correlation between the two batches
 - Essentially solving SVD of $Y_1^T Y_2$
 - Left and right eigenvectors of $Y_1^T Y_2$ are estimates of W_1 and W_2
 - Treat CCA as a dimension reduction step that minimize the effect of batches
 - Why CCA instead of PCA?



Seurat CCA

- MultiCCA (v1) (Butler et. al. Nature Biotech 2018) further uses dynamic time wrapping to further align the CC vectors to remove remaining batch effects



- They later have developed multiCCA (v2) which is hybrid between multiCCA (v1) and MNNcorrect

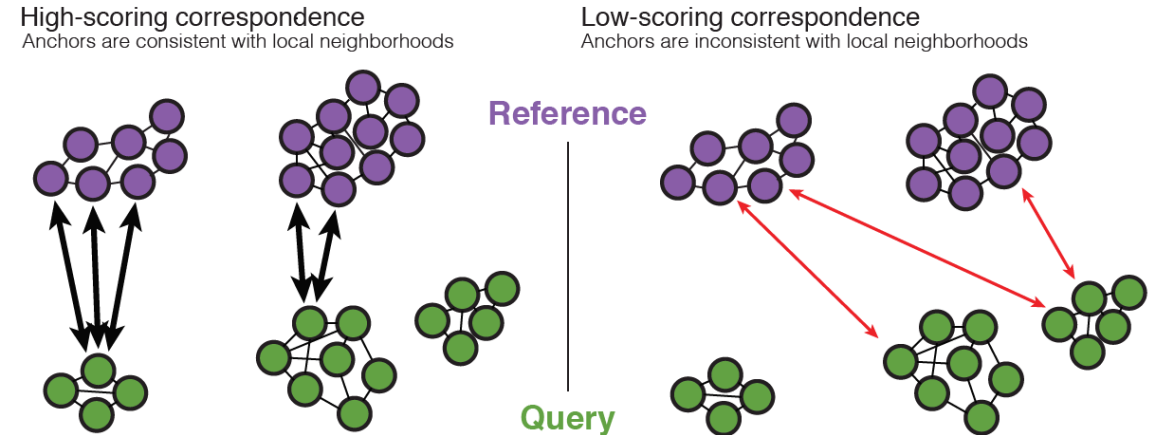
MultiCCA v2 (Stuart et. al. Cell, 2019)

- Steps

- CCA as in MultiCCA V1 to project both datasets into lower dimensions
 - PCA may amplify differences between two datasets and focus on variation directions that are unique to one dataset

- Identify anchor cells using MNN

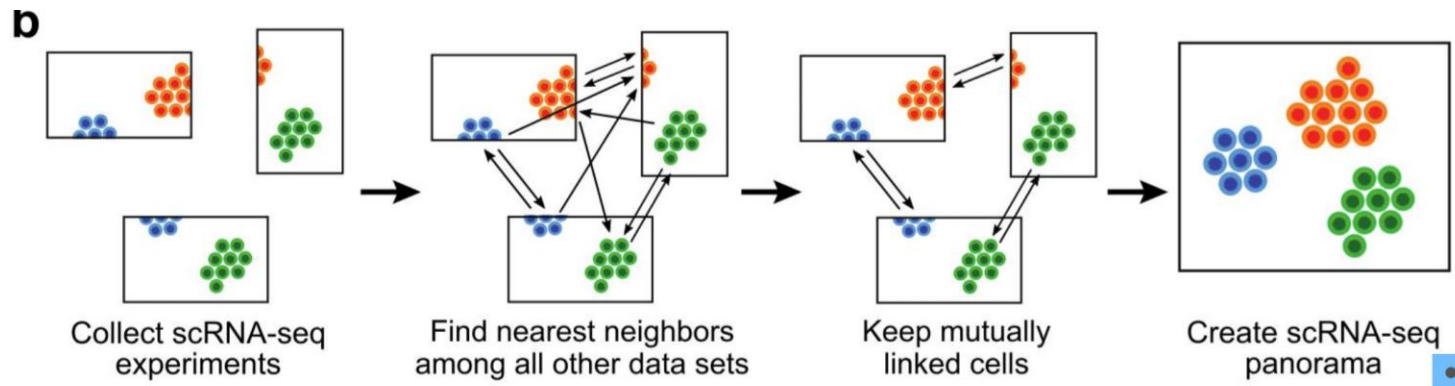
- Give each cell an anchor score
 - Check MNN also in the original space to improve robustness
 - **Anchor scoring**: find consistency of KNNs within each dataset and with other datasets
 - **Anchor weighting W** : a matrix of anchors by cells in Y_2
 - Weights depend on cell-cell distance, only use k nearest anchors
 - **Alignment**: $\hat{Y}_2 = Y_2 + (Y_{1,A} - Y_{2,A})W$
 - Multiple datasets: align sequentially



Scanorama (Hie et. al. , Nature Biotech 2019)

Main advantage: computationally fast MNN

- Find KNN of a cell in one dataset from all other datasets
 - To reduce computational cost in finding KNN by approximation with random projection trees to make computational cost less than $O(kn)$
- Anchor cells: keep a pair of cells if they are KNN to each other
 - Computational cost reduce from $O(k^2 n_1 n_2)$ to $O(k \min(n_1, n_2))$
- Batch correction from anchors using Gaussian kernel smoothing weights same as MNNcorrect/fastMNN
- Scanorama performs better than MNNcorrect/fastMNN in benchmarking studies
 - Only methodological difference between Scanorama and MNNcorrect/fastMNN seems to be the dimension reduction first step before finding KNN



Harmony (Korsunsky et. al. Nature Methods, 2019)

Steps

- PCA
- Iteratively perform
 - Soft k-means clustering
 - Penalize clusters that has less batch diversity

$$\min_{R,Y} \sum_{i,k} R_{ki} \|Z_i - Y_k\|^2 + \sigma R_{ki} \log R_{ki} + \sigma \theta R_{ki} \log \left(\frac{O_{ki}}{E_{ki}} \right) \phi_i$$

$$\text{s. t. } \forall_i \forall_k R_{ki} > 0, \forall_i \sum_{k=1}^K R_{ki} = 1$$

$O \in [0, 1]^{K \times B}$ The observed co-occurrence matrix of cells in clusters (rows) and batches (columns).

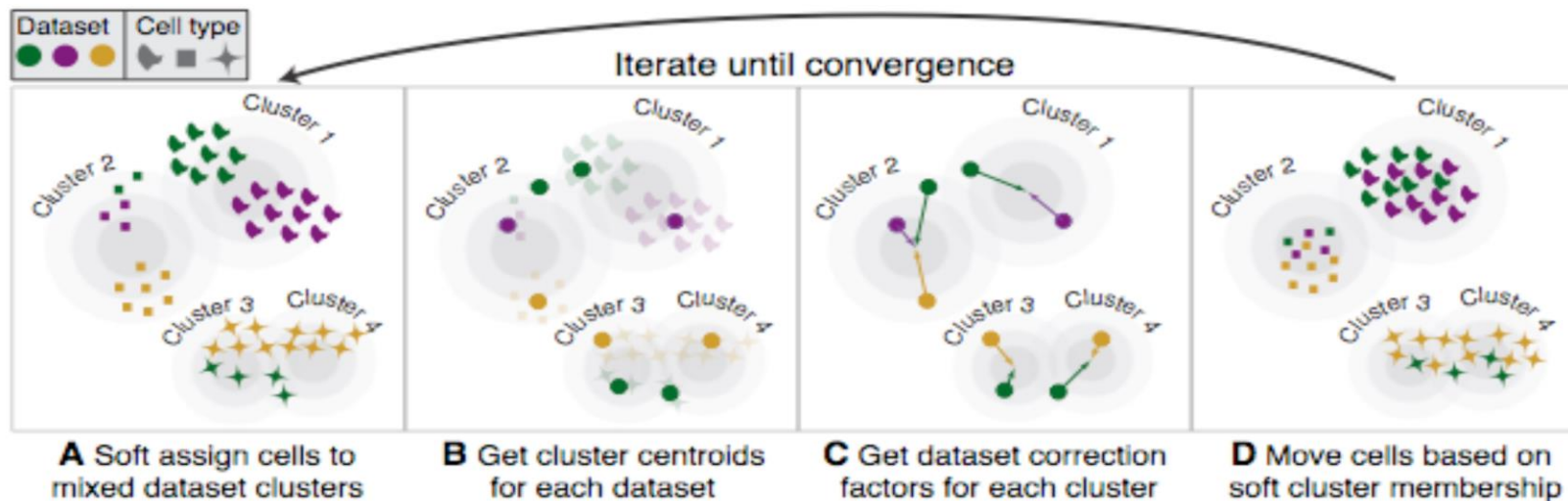
$E \in [0, 1]^{K \times B}$ The expected co-occurrence matrix of cells in clusters and batches, under the assumption of independence between cluster and batch assignment.

$Y \in [0, 1]^{d \times K}$ Cluster centroid locations in the k -means clustering algorithm.

Harmony (Korsunsky et. al. Nature Methods, 2019)

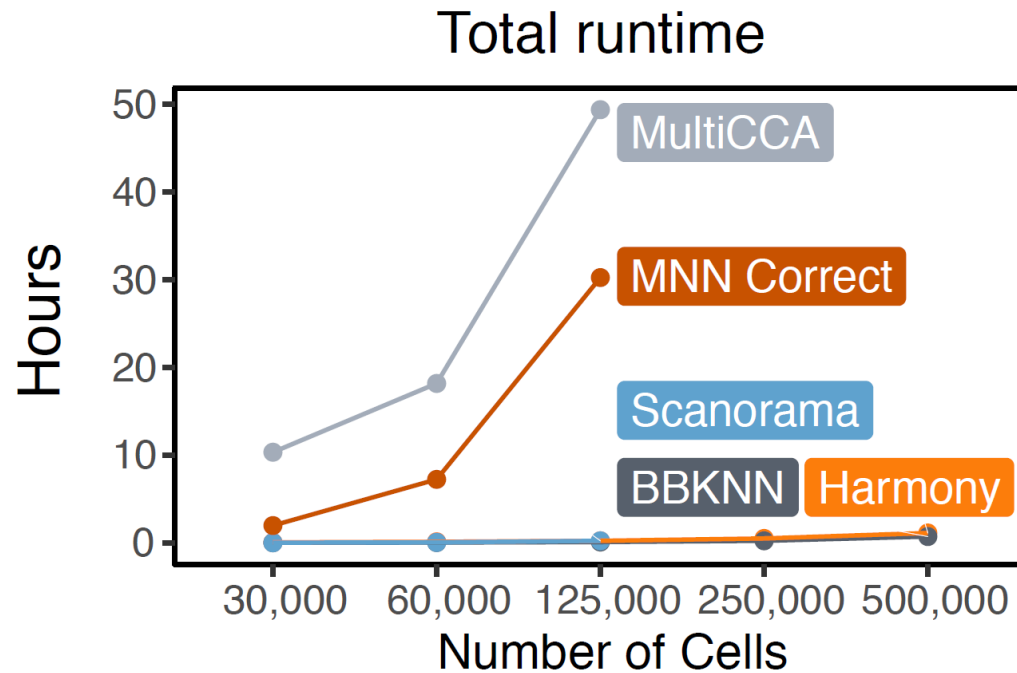
Steps

- PCA
- Iteratively perform
 - Soft k-means clustering
 - Penalize clusters that has less batch diversity
 - Goal: make cells of the same cell type in each cluster
 - Mixture of experts model for correction
 - Compute cluster-specific batch correction by linear regression
 - Assume that the mean of each cell within each cluster linearly depend on the batch information
 - Move cells in each cluster by subtracting the batch and cluster specific mean effect

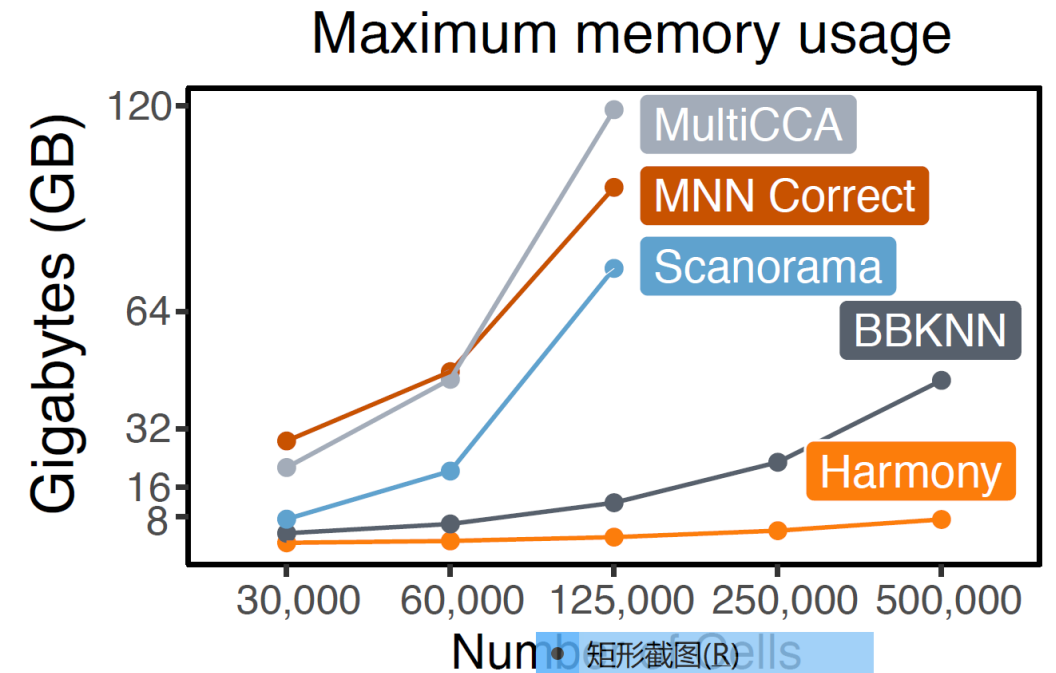


Comparison of computational costs

A



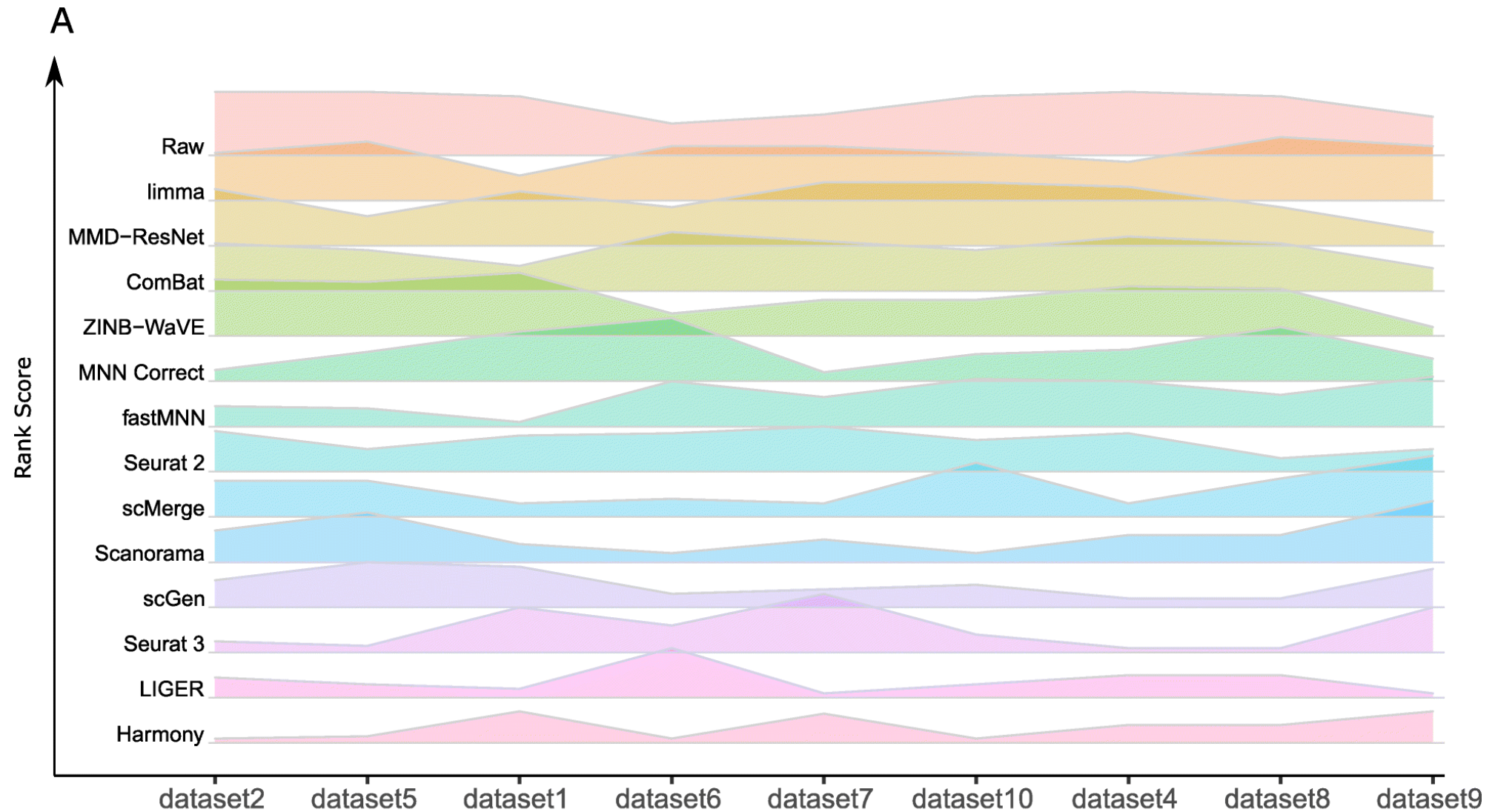
B



Summary















































































































































































































- Factor-model-based methods and cell-similarity based methods seem to be based on two different sets of assumptions on the batch effects
 - Factor-model-based: batch information and latent factors are nearly orthogonal to each other
 - Cell-similarity based: batch effect is very small compared to biological signals
 - The two assumptions seem quite different -> what is the consequence on performance?
- Factor-model-based methods can provide batch corrected gene expression matrix
 - May introduce false positives in down-stream differential testing
- Performance: Without using additional cell type information, cell-similarity based methods perform slightly better but the two strategies seem comparable

Benchmarking results (Tran et. al. Genome Biology 2020)



Benchmarking results (Luecken et. al. Nature Methods 2022)

b

Method					RNA					Simulations		Usability		Scalability	
Rank	Name	Output	Features	Scaling	Pancreas	Lung	Immune (human)	Immune (human/mouse)	Mouse brain	Sim 1	Sim 2	Package	Paper	Time	Memory
1	scANVI*		HVG	-		2	3		1	1	2				3
2	Scanorama		HVG	+			1	2		2					
3	scVI		HVG	-		3		3							1
4	fastMNN		HVG	-			2				3				
5	scGen*		HVG	-	3	1		1			1				
6	Harmony		HVG	-	1								1		
7	fastMNN		HVG	-											
8	Seurat v3 RPCA		HVG	+	2							1			
9	BBKNN		HVG	-					2				3	2	2
10	Scanorama		HVG	+											
11	ComBat		HVG	-					3			3		1	
12	MNN		HVG	+											
13	Seurat v3 CCA		HVG	-								1			
14	trVAE		HVG	-											
15	Conos		HVG	-									1		
16	DESC		FULL	-						3					
17	LIGER		HVG	-											
18	SAUCIE		HVG	+										3	
19	Unintegrated		FULL	-											
20	SAUCIE		HVG	+										3	

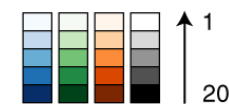
Output

- Genes
- Embedding
- Graph

Scaling

- + Scaled
- Unscaled

Ranking



Related papers

- Argelaguet, R., Cuomo, A. S., Stegle, O., & Marioni, J. C. (2021). Computational principles and challenges in single-cell data integration. *Nature biotechnology*, 39(10), 1202-1215.
- Ritchie, M. E., Phipson, B., Wu, D. I., Hu, Y., Law, C. W., Shi, W., & Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic acids research*, 43(7), e47-e47.
- Risso, D., Perraudeau, F., Gribkova, S., Dudoit, S., & Vert, J. P. (2018). A general and flexible method for signal extraction from single-cell RNA-seq data. *Nature communications*, 9(1), 284.
- Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., & Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. *Nature methods*, 15(12), 1053-1058.
- Lotfollahi, M., Wolf, F. A., & Theis, F. J. (2019). scGen predicts single-cell perturbation responses. *Nature methods*, 16(8), 715-721.
- Xu, C., Lopez, R., Mehlman, E., Regier, J., Jordan, M. I., & Yosef, N. (2021). Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. *Molecular systems biology*, 17(1), e9620.
- Tran, H. T. N., Ang, K. S., Chevrier, M., Zhang, X., Lee, N. Y. S., Goh, M., & Chen, J. (2020). A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome biology*, 21, 1-32.
- Haghverdi, L., Lun, A. T., Morgan, M. D., & Marioni, J. C. (2018). Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nature biotechnology*, 36(5), 421-427.
- Butler, A., Hoffman, P., Smibert, P., Papalexi, E., & Satija, R. (2018). Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology*, 36(5), 411-420.
- Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck, W. M., ... & Satija, R. (2019). Comprehensive integration of single-cell data. *cell*, 177(7), 1888-1902.
- Hie, B., Bryson, B., & Berger, B. (2019). Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. *Nature biotechnology*, 37(6), 685-691.
- Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., ... & Raychaudhuri, S. (2019). Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature methods*, 16(12), 1289-1296.
- Luecken, M. D., Büttner, M., Chaichoompu, K., Danese, A., Interlandi, M., Müller, M. F., ... & Theis, F. J. (2022). Benchmarking atlas-level data integration in single-cell genomics. *Nature methods*, 19(1), 41-50.